VHDL for Synthesis

Course Description

This course provides all necessary theoretical and practical know-how to write an efficient synthesizable HDL code through VHDL standard language.

The course goes into great depth and teaches efficient methods for writing VHDL code in a way that produces the precise digital circuit for various constraints like high frequency, low power, and minimal area.

The course covers the full synthesis process flow starting from reviewing methodologies, using development tools, adding constraints, implementing every VHDL structure in an optimal way, understanding the problems with bad coding style, learning the differences between simulation pre- and post-synthesis, analyzing critical paths, and reading and analyzing synthesis reports.

In addition, the course focuses on writing efficient code to save area, increasing frequency, designing for low power consumption, dealing with skew problems, working with external IPs, using attributes in VHDL code, implementing reliable, and high-speed finite state machines, solving design problems like high fanout and more.

The course combines 50% theory with 50% practical work in every meeting. The practical labs cover all the theory and also include practical digital design.

Course Duration

5 days

Goals

1. Understand the synthesis process flow and the difference between different tools
2. Learn precise coding style for combinational and sequential circuits
3. Understand synthesis of multi-files projects
4. Become familiar with the various constraints and adding them to project
5. Become familiar with design problems resulting from bad coding style
6. Design efficient circuits for minimal area or high frequency
7. Work with IPs and combining them in the synthesis flow
8. Produce reports, detect and correct timing problems

Intended Users

Hardware engineers who develop FPGAs and would like to enhance their skills, in order to understand synthesis limitations, to acquire better expertise on avoiding digital problems and to be to write efficient coding style for synthesis

Previous Knowledge

FPGA design, VHDL

Course Material

1. Simulator: Modelsim
2. Synthesizer and Place & Route: Quartus Prime
3. Course book (including labs)

Table of Contents

Day #1

        - LATCH problem
        - Don't care in synthesis
        - std_match function
        - WITH-SELECT synthesis
    - Inference from arithmetic and relational operators
        - Integer versus REAL
        - Arithmetic operators: + , -, *, /, abs, **
        - Relational operators: >, <, =, /=, >=, <=
        - Constants in arithmetic and relational operators
    - Simulation versus synthesis behavior and synthesis guidelines

- **Operator Sharing**
    - Derivation of efficient HDL description
    - Reducing circuit size
    - Operator sharing using VHDL description
    - Operator sharing using synthesizer GUI options
    - Analyzing area and frequency
    - Tradeoff analysis
    - Complex operator sharing and synthesis tools limitations

Day #2

- **Sequential Statements Synthesis**
    - Inference from within processes introduction
    - General guidelines
        - Using variables
        - Case versus IF-ELSE
        - Combinational sensitivity list
        - Sequential sensitivity list
    - Inference from simple assignment statements
        - Signals versus Variables synthesis
    - Inference from IF-THEN-ELSE and IF-THEN-ELSIF statements
        - Priority
        - Latch problem
        - Full versus partial sensitivity list
        - Combinational versus sequential If statements

- Variables versus Signals synthesis
- Operator sharing within process
- Nested IF statements synthesis
- Analyzing results on different synthesizers

- Inference from Case statements
  - When-Others statement synthesis
  - Combinational versus sequential sensitivity list
  - Null key word
  - Latch problem
  - Don't care in Case statements
- Inference from loop statements
  - Serial loop versus parallel loop synthesis
  - Loop index
  - FOR loop versus while and simple loop synthesis
  - Variables versus Signals in loop synthesis
- Combinational circuit design examples
  - Gray code incrementor
  - Programmable priority encoder
  - Signed addition with status
  - Combinational adder based multiplier
  - Hamming distance circuit

Day #3

- **Sequential Statements Synthesis (continue)**
  - Incomplete Sensitivity List
    - Bad versus good coding style, simulation pre and post synthesis
  - Inference using Variables Versus Signals
    - variables and signals in combinational processes, variables and signals in sequential processes, when variable becomes register, analyzing synthesis warnings, debugging variables in post synthesis simulation
  - Synchronous Circuits Inference
    - latch versus flip-flop inference
    - Synchronous and asynchronous reset
    - Load and enable signals

- Inference from WAIT Statements
  - Combinational WAIT processes
  - Sequential WAIT processes
  - Non synthesizable WAIT statements
  - Variables versus signals in WAIT statements
  - Synchronous reset in WAIT statements

- **Finite State Machine Synthesis**
  - State machine structure (Mealy and Moore block diagram)
  - State encoding (Auto, one-hot, binary, Johnson, two-hot, Gray, User specific)
  - State machine implementation in VHDL (one, two or three processes)
  - Bad coding style for state machine
  - Specifying encoding style in VHDL and in synthesizer tool
  - Analyzing encoding style area and performance
  - High reliability safe state machines using attributes, handling illegal states

Day #4

- **Timing Analysis of a Synchronous Sequential Circuits**
  - Introduction to timing constraints and synchronous design techniques
  - Synchronized versus unsynchronized I/O
  - Setup time violation and maximal clock rate
  - Synthesis static analysis formula
  - Hold time violation
  - Output related timing considerations
  - Input related timing considerations
  - Poor design practices and their remedies
    - Misuse of asynchronous signals
    - Misuse of gated clock
    - Misuse of derived clocks
    - Global clock
    - Clock skew
  - Analyzing critical paths in details
  - Synthesis static timing analysis versus Place & Route static timing analysis

- Fanout and long combinational chain timing problems
- Using PLL in the system
- Physical Synthesis versus logical synthesis

Day #5

- **Synthesis of Large Projects**
  - Synthesis of Package & Package Body
  - Synthesis of functions & procedures
  - Reuse methodology
  - Synthesis of IEEE packages

- **Integrating IP Core**
  - IP core generation and integration into VHDL code
  - IP black box constraints

- **Maximize Clock Rate**
  - Introduction to pipeline
  - Latency and throughput
  - Pipelined combinational circuits
  - Pipelining considerations
  - Pipeline balancing
  - Effectiveness of pipeline
  - Adding a pipeline in VHDL
  - Analyze clock rate in pipelined design
  - Complex pipeline circuits
  - Retiming